PROBLEM. Secret (contributed by N. R. Lim-Cheng) [Score: 50/50]

In Computer Science, the process of sending secret messages to ensure confidentiality is called encryption. *Encryption* is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. There are many encryption algorithms available. One algorithm is via the use of the same key (known as the secret key) to encrypt and decrypt the message. The following is a sample of the partial contents of a secret key:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
'e'	't'	'u'	'm'	'y'	'b'	'c'	'd'	'f'	'g'	'h'	'l'	'j'	'k'	'l'	'n'	'o'	'p'	'q'	'r'	's'	'v'	'w'	'x'	'z'	'a'	'\0'

We will encrypt 'a' with the first element in the secret key (in the example, it is 'e'), encrypt 'b' with the second element in the secret key (in the example, it is 't'), encrypt 'c' with the third element in the secret key (in the example, it is 'u'), and so on. So, if the message is a string "zea", it should be encrypted into "aye".

Consequently, if the encrypted message contains the string "aye", it is decrypted into "zea". Here's another example: If the encrypted message contains the string "dazbe", the decrypted message is "meaty".

Assume that the characters in the input message to be encrypted are all small letters. Thus, the characters in the resulting decrypted message will also be all small letters. The secret key will always contain unique characters.

Hint: 'a' is equivalent to 97 in ASCII, 'b' is 98, and so on. The space character ' ' is equivalent to 32 in ASCII.

For this problem, you need to implement the functions for each of the following features:

1.) Encrypt [10 pts]– this is to encode a word using the secret key. Refer to the examples above.
2.) Decrypt [15 pts] – this is to decode the hidden word using the secret key. Refer to the examples above.
3.) Tokenize [15 pts]– this is to split the given text (consisting of letters and spaces) into an array of words.
4.) Merge [10 pts]– this is to put together the array of words into 1 long text separated by exactly 1 space between words. Note that there should be no space after the last word.

General Instructions and Restrictions:

1. Use and follow the given file templates/skeleton files.
2. Your solution should NOT include any input or output statements (e.g., scanf() or printf())
3. You are only allowed to use strlen(), strcmp(), strcat(), and strcpy() from string.h. No other functions from string.h can be used.
4. You are not allowed to include other header files in the templates, other than those indicated.
5. Your submitted file LASTNAME-secret.c should not contain main().

You are given five files for this problem:

(1) `secret.h` which contains the type declaration and function prototypes that will be used in the program. This file cannot be modified.
(2) `LASTNAME-secret.c` which contains some initial code that you'll need to complete. Comments indicate the requirement and restrictions imposed for the solution. For this file, the student's implementation should not include any scanf) or printf(). This file should not be modified to contain the main().
(3) `secretMain.c` which contains the main() that can be used to test the requirement
(4) `INPUT.txt` contains all the sequence of [sample] inputs needed by secretMain.c
(5) `EXPECTED.txt` contains the expected output of running the executable file generated from secretMain.c and using the inputs from INPUT.txt. For ease in debugging and testing, it is recommended that the student use input and output redirection. Assuming you have a.exe as the executable file from compiling secretMain.c, to do input and output redirection, type the following in the command prompt (in the same folder where the a.exe and the INPUT.txt are):
For Windows: `a < INPUT.txt > LASTNAME-ACTUAL.txt`

You can then open LASTNAME-ACTUAL.txt and compare the contents to EXPECTED.txt. Note that even the spacing should be exactly the same.

DELIVERABLE: Your C program source file **`LASTNAME-secret.c`** with your own last name as filename. For example, if your lastname is SANTOS, then the source file should be named as `SANTOS-secret.c`. Upload your source file in AnimoSpace before the indicated submission time.

TESTING & SCORING:

- Your program will be compiled via gcc -Wall, using C99 standard. Thus, for each function that does not compile successfully, the score for that function is 0.
- Your program will be tested by your instructor with other TEXT FILES (contents are different from the ones given to you) and with function calls of different parameter values.
- Full credit will be given for the function only if the student's implementation are all correct for all the test values used by the instructor during checking AND only if the student's implementation complied with the requirement and did not violate restrictions. Deductions will be given if not all test cases produce correct results OR if restrictions were not followed.